Intrinsic complexity in arithmetic (and algebra)

Yiannis N. Moschovakis
UCLA and University of Athens

JAF32, Athens, June 26, 2013

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\}$, $a \geq b \geq 1$,

$$(\varepsilon) \quad \boxed{\mathrm{gcd}(a, b) = \text{if } (\mathrm{rem}(a, b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a, b))}$$

where $a = \mathrm{iq}(a, b)b + \mathrm{rem}(a, b) \quad (0 \leq \mathrm{rem}(a, b) < b)$

$$\begin{aligned}\mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a, b)\\ &\leq 2\log(b) \qquad (a \geq b \geq 2)\end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a, b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp b \iff \mathrm{gcd}(a, b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \ \ \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a))$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\}$, $a \geq b \geq 1$,

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a,b) = \text{if } (\mathrm{rem}(a,b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a,b))}$$

where $a = \mathrm{iq}(a,b)b + \mathrm{rem}(a,b) \quad (0 \leq \mathrm{rem}(a,b) < b)$

$$\mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) = \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a,b)$$
$$\leq 2\log(b) \qquad (a \geq b \geq 2)$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a,b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp b \iff \mathrm{gcd}(a,b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a))$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\}$, $a \geq b \geq 1$,

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a, b) = \text{if } (\mathrm{rem}(a, b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a, b))}$$

where $a = \mathrm{iq}(a, b)b + \mathrm{rem}(a, b) \quad (0 \leq \mathrm{rem}(a, b) < b)$

$$\begin{aligned} \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a, b) \\ &\leq 2\log(b) \qquad (a \geq b \geq 2) \end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a, b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp b \iff \mathrm{gcd}(a, b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a))$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\},\ a \geq b \geq 1,$

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a, b) = \text{if } (\mathrm{rem}(a, b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a, b))}$$

where $a = \mathrm{iq}(a, b)b + \mathrm{rem}(a, b) \quad (0 \leq \mathrm{rem}(a, b) < b)$

$$\begin{aligned} \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a, b) \\ &\leq 2\log(b) \qquad (a \geq b \geq 2) \end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a, b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp b \iff \mathrm{gcd}(a, b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a))$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\}$, $a \geq b \geq 1$,

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a, b) = \text{if } (\mathrm{rem}(a, b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a, b))}$$

where $a = \mathrm{iq}(a, b)b + \mathrm{rem}(a, b) \quad (0 \leq \mathrm{rem}(a, b) < b)$

$$\begin{aligned}\mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a, b)\\ &\leq 2\log(b) \qquad (a \geq b \geq 2)\end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a, b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp\!\!\!\perp b \iff \mathrm{gcd}(a, b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$

- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a)$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\},\ a \geq b \geq 1$,

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a,b) = \text{if } (\mathrm{rem}(a,b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a,b))}$$

where $a = \mathrm{iq}(a,b)b + \mathrm{rem}(a,b) \quad (0 \leq \mathrm{rem}(a,b) < b)$

$$\begin{aligned} \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a,b) \\ &\leq 2\log(b) \qquad (a \geq b \geq 2) \end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a,b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp\!\!\!\perp b \iff \mathrm{gcd}(a,b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a))$$

Is the Euclidean algorithm optimal from its primitives?

For $a, b \in \mathbb{N} = \{0, 1, \ldots\}, \ a \geq b \geq 1$,

$$(\varepsilon) \qquad \boxed{\mathrm{gcd}(a,b) = \text{if } (\mathrm{rem}(a,b) = 0) \text{ then } b \text{ else } \mathrm{gcd}(b, \mathrm{rem}(a,b))}$$

where $a = \mathrm{iq}(a,b)b + \mathrm{rem}(a,b) \quad (0 \leq \mathrm{rem}(a,b) < b)$

$$\begin{aligned} \mathrm{calls}_{\{\mathrm{rem}\}}(\varepsilon, a, b) &= \text{the number of divisions } \varepsilon \text{ needs to compute } \mathrm{gcd}(a,b) \\ &\leq 2\log(b) \qquad (a \geq b \geq 2) \end{aligned}$$

- $\boxed{\text{Is } \varepsilon \text{ optimal for computing } \mathrm{gcd}(a,b) \text{ from } \{\mathrm{rem}, =_0\}?}$
- $a \perp\!\!\!\perp b \iff \mathrm{gcd}(a,b) = 1$

  $\boxed{\text{Is } \varepsilon \text{ optimal for deciding coprimeness from } \{\mathrm{rem}, =_0, =_1\}?}$
- And is this true for all algorithms from $\{\mathrm{rem}, =_0, =_1\}$?

Conjecture: *For every algorithm α which decides coprimeness from $\{\mathrm{rem}, =_0, =_1\}$*

$$(\exists r > 0)(\text{for infinitely many } a \geq b, \quad \mathrm{calls}_{\{\mathrm{rem}\}}(\alpha, a, b) \geq r\log(a)$$

The value complexities I

• A classical method for establishing lower bounds that restrict all algorithms assuming practically nothing about "what algorithms are":

Horner's rule: For any field F and $n \geq 1$, the value of a polynomial of degree n can be computed using no more than n multiplications and n additions in F:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = a_0 + x\Big(a_1 + a_2 x + \cdots + a_n x^{n-1}\Big)$$

Theorem (Pan 1966, (Winograd 1967, 1970))

Every algorithm from the complex field operations requires at least n multiplications/divisions and at least n additions/subtractions to compute $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ when $\vec{a}, x$ are algebraically independent complex numbers (the generic case)

... because it takes that many applications of the field operations to construct the value $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ from $a_0, \ldots, a_n, x$

The value complexities I

• A classical method for establishing lower bounds that restrict all algorithms assuming practically nothing about "what algorithms are":

Horner's rule: For any field F and $n \geq 1$, the value of a polynomial of degree n can be computed using no more than n multiplications and n additions in F:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = a_0 + x\Big(a_1 + a_2 x + \cdots + a_n x^{n-1}\Big)$$

Theorem (Pan 1966, (Winograd 1967, 1970))

Every algorithm from the complex field operations requires at least n multiplications/divisions and at least n additions/subtractions to compute $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ when $\vec{a}, x$ are algebraically independent complex numbers (the generic case)

...because it takes that many applications of the field operations to construct the value $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ from $a_0, \ldots, a_n, x$

The value complexities I

• A classical method for establishing lower bounds that restrict all algorithms assuming practically nothing about "what algorithms are":

Horner's rule: For any field F and $n \geq 1$, the value of a polynomial of degree n can be computed using no more than n multiplications and n additions in F:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = a_0 + x\Big(a_1 + a_2 x + \cdots + a_n x^{n-1}\Big)$$

Theorem (Pan 1966, (Winograd 1967, 1970))

Every algorithm from the complex field operations requires at least n multiplications/divisions and at least n additions/subtractions to compute $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ when $\vec{a}, x$ are algebraically independent complex numbers (the generic case)

…because it takes that many applications of the field operations to construct the value $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ from $a_0, \ldots, a_n, x$

The value complexities I

• A classical method for establishing lower bounds that restrict all algorithms assuming practically nothing about "what algorithms are":

Horner's rule: For any field F and $n \geq 1$, the value of a polynomial of degree n can be computed using no more than n multiplications and n additions in F:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = a_0 + x\Big(a_1 + a_2 x + \cdots + a_n x^{n-1}\Big)$$

Theorem (Pan 1966, (Winograd 1967, 1970))

Every algorithm from the complex field operations requires at least n multiplications/divisions and at least n additions/subtractions to compute $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ when $\vec{a}, x$ are algebraically independent complex numbers (the generic case)

...because it takes that many applications of the field operations to construct the value $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ from $a_0, \ldots, a_n, x$

The value complexities II

Theorem (van den Dries)

If an algorithm α *computes* $\mathrm{gcd}(x, y)$ *from* $0, 1, +, -, \mathrm{iq}, \mathrm{rem}, \cdot, <$ *and*

$$\mathrm{calls}(\alpha, x, y) = \text{ the number of calls to the primitives } \alpha \text{ makes to compute } \mathrm{gcd}(x, y),$$

then for all $a > b$ *such that* $a^2 = 2b^2 + 1$ (Pell pairs),

$$\mathrm{calls}(\alpha, a + 1, b) \geq \frac{1}{4}\sqrt{\log \log b}$$

... because it takes at least that many applications of the primitives to construct the value $\mathrm{gcd}(a + 1, b)$ when (a, b) is a Pell pair

- *This method cannot yield lower bounds for decision problems* (because their output (tt or ff) is available with no computation)
- and it is open whether algorithms that decide coprimeness from these primitives (which include multiplication) must execute $O(\sqrt{\log \log \max(x, y)})$ operations on an infinite set of inputs

The value complexities II

Theorem (van den Dries)

If an algorithm α *computes* $\mathrm{gcd}(x, y)$ *from* $0, 1, +, -, \mathrm{iq}, \mathrm{rem}, \cdot, <$ *and*

$\mathrm{calls}(\alpha, x, y) =$ *the number of calls to the primitives*
α *makes to compute* $\mathrm{gcd}(x, y)$,

then for all $a > b$ *such that* $a^2 = 2b^2 + 1$ (Pell pairs),

$$\mathrm{calls}(\alpha, a + 1, b) \geq \frac{1}{4}\sqrt{\log \log b}$$

. . . because it takes at least that many applications of the primitives to construct the value $\mathrm{gcd}(a + 1, b)$ when (a, b) is a Pell pair

- *This method cannot yield lower bounds for decision problems* (because their output (tt or ff) is available with no computation)
- and it is open whether algorithms that decide coprimeness from these primitives (which include multiplication) must execute $O(\sqrt{\log \log \max(x, y)})$ operations on an infinite set of inputs

The value complexities II

Theorem (van den Dries)

If an algorithm α *computes* $\text{gcd}(x, y)$ *from* $0, 1, +, -, \text{iq}, \text{rem}, \cdot, <$ *and*

$$\text{calls}(\alpha, x, y) = \textit{ the number of calls to the primitives } \alpha \textit{ makes to compute } \text{gcd}(x, y),$$

then for all $a > b$ *such that* $a^2 = 2b^2 + 1$ (Pell pairs),

$$\text{calls}(\alpha, a + 1, b) \geq \frac{1}{4}\sqrt{\log \log b}$$

. . . because it takes at least that many applications of the primitives to construct the value $\text{gcd}(a + 1, b)$ when (a, b) is a Pell pair

- *This method cannot yield lower bounds for decision problems* (because their output (tt or ff) is available with no computation)
- and it is open whether algorithms that decide coprimeness from these primitives (which include multiplication) must execute $O(\sqrt{\log \log \max(x, y)})$ operations on an infinite set of inputs

The value complexities II

Theorem (van den Dries)

If an algorithm α *computes* $\mathrm{gcd}(x, y)$ *from* $0, 1, +, -, \mathrm{iq}, \mathrm{rem}, \cdot, <$ *and*

$$\mathrm{calls}(\alpha, x, y) = \textit{ the number of calls to the primitives } \alpha \textit{ makes to compute } \mathrm{gcd}(x, y),$$

then for all $a > b$ *such that* $a^2 = 2b^2 + 1$ (Pell pairs),

$$\mathrm{calls}(\alpha, a+1, b) \geq \frac{1}{4}\sqrt{\log \log b}$$

...because it takes at least that many applications of the primitives to construct the value $\mathrm{gcd}(a + 1, b)$ when (a, b) is a Pell pair

- *This method cannot yield lower bounds for decision problems* (because their output (tt or ff) is available with no computation)
- and it is open whether algorithms that decide coprimeness from these primitives (which include multiplication) must execute $O(\sqrt{\log \log \max(x, y)})$ operations on an infinite set of inputs

(Partial) structures

- A (partial) structure is a tuple $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ where Φ is a set of function and relation symbols and $\Phi^{\mathbf{A}} = \{\phi^{\mathbf{A}}\}_{\phi \in \Phi}$, where with $s_\phi \in \{\mathtt{a}, \mathtt{boole}\}$, $A_{\mathtt{a}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}$,

$$\boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A_{s_\phi}} \text{ i.e., } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A} \text{ or } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}}$$

- $\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =)$, the standard structure of arithmetic
- $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, =_0, =_1)$, the Euclidean structure
- $\mathbf{N}_\varepsilon \upharpoonright U = (U, \mathrm{rem} \upharpoonright U, =_0 \upharpoonright U, =_1 \upharpoonright U)$ where $U \subseteq \mathbb{N}$ and

$$(f \upharpoonright U)(x, y) = w \iff \vec{x} \in U^n, w \in U_s \ \& \ f(\vec{x}) = w$$

- The (equational) diagram of a Φ-structure is the set of its basic equations,

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \vec{x} \in A^{n_\phi}, w \in A_{s_\phi}, \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\}$$

- We may assume that $\mathbf{A}$ is completely determined by $\mathrm{eqdiag}(\mathbf{A})$

(Partial) structures

- A (partial) structure is a tuple $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ where Φ is a set of function and relation symbols and $\Phi^{\mathbf{A}} = \{\phi^{\mathbf{A}}\}_{\phi\in\Phi}$, where with $s_\phi \in \{\mathtt{a}, \mathtt{boole}\}$, $A_{\mathtt{a}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}$,

$$\boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A_{s_\phi}} \text{ i.e., } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A} \text{ or } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}}$$

- $\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =)$, the standard structure of arithmetic
- $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, =_0, =_1)$, the Euclidean structure
- $\mathbf{N}_\varepsilon \upharpoonright U = (U, \mathrm{rem} \upharpoonright U, =_0 \upharpoonright U, =_1 \upharpoonright U)$ where $U \subseteq \mathbb{N}$ and

$$(f \upharpoonright U)(x, y) = w \iff \vec{x} \in U^n, w \in U_s \;\&\; f(\vec{x}) = w$$

- The (equational) diagram of a Φ-structure is the set of its basic equations,

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \vec{x} \in A^{n_\phi}, w \in A_{s_\phi}, \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\}$$

- We may assume that $\mathbf{A}$ is completely determined by $\mathrm{eqdiag}(\mathbf{A})$

(Partial) structures

- A (partial) structure is a tuple $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ where Φ is a set of function and relation symbols and $\Phi^{\mathbf{A}} = \{\phi^{\mathbf{A}}\}_{\phi \in \Phi}$, where with $s_\phi \in \{\mathtt{a}, \mathtt{boole}\}$, $A_{\mathtt{a}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}$,

$$\boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A_{s_\phi}} \text{ i.e., } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A} \text{ or } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}}$$

- $\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =)$, the standard structure of arithmetic
- $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, =_0, =_1)$, the Euclidean structure
- $\mathbf{N}_\varepsilon \restriction U = (U, \mathrm{rem} \restriction U, =_0 \restriction U, =_1 \restriction U)$ where $U \subseteq \mathbb{N}$ and

$$(f \restriction U)(x, y) = w \iff \vec{x} \in U^n, w \in U_s \ \&\ f(\vec{x}) = w$$

- The (equational) diagram of a Φ-structure is the set of its basic equations,

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \vec{x} \in A^{n_\phi}, w \in A_{s_\phi}, \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\}$$

- We may assume that $\mathbf{A}$ is completely determined by $\mathrm{eqdiag}(\mathbf{A})$

(Partial) structures

- A (partial) structure is a tuple $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ where Φ is a set of function and relation symbols and $\Phi^{\mathbf{A}} = \{\phi^{\mathbf{A}}\}_{\phi \in \Phi}$, where with $s_\phi \in \{\mathtt{a}, \mathtt{boole}\}$, $A_{\mathtt{a}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}$,

$$\boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A_{s_\phi}} \text{ i.e., } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A} \text{ or } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}}$$

- $\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =)$, the standard structure of arithmetic
- $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, =_0, =_1)$, the Euclidean structure
- $\mathbf{N}_\varepsilon \restriction U = (U, \mathrm{rem} \restriction U, =_0 \restriction U, =_1 \restriction U)$ where $U \subseteq \mathbb{N}$ and

$$(f \restriction U)(x, y) = w \iff \vec{x} \in U^n, w \in U_s \,\&\, f(\vec{x}) = w$$

- The (equational) diagram of a Φ-structure is the set of its basic equations,

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \vec{x} \in A^{n_\phi}, w \in A_{s_\phi}, \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\}$$

- We may assume that $\mathbf{A}$ is completely determined by $\mathrm{eqdiag}(\mathbf{A})$

(Partial) structures

- A (partial) structure is a tuple $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ where Φ is a set of function and relation symbols and $\Phi^{\mathbf{A}} = \{\phi^{\mathbf{A}}\}_{\phi \in \Phi}$, where with $s_\phi \in \{\mathtt{a}, \mathtt{boole}\}$, $A_{\mathtt{a}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}$,

$$\boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A_{s_\phi}} \text{ i.e., } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup A} \text{ or } \boxed{\phi^{\mathbf{A}} : A^{n_\phi} \rightharpoonup \{\mathrm{tt}, \mathrm{ff}\}}$$

- $\mathbf{N} = (\mathbb{N}, 0, 1, +, \cdot, =)$, the standard structure of arithmetic
- $\mathbf{N}_\varepsilon = (\mathbb{N}, \mathrm{rem}, =_0, =_1)$, the Euclidean structure
- $\mathbf{N}_\varepsilon \restriction U = (U, \mathrm{rem} \restriction U, =_0 \restriction U, =_1 \restriction U)$ where $U \subseteq \mathbb{N}$ and

$$(f \restriction U)(x, y) = w \iff \vec{x} \in U^n, w \in U_s \ \&\ f(\vec{x}) = w$$

- The (equational) diagram of a Φ-structure is the set of its basic equations,

$$\mathrm{eqdiag}(\mathbf{A}) = \{(\phi, \vec{x}, w) : \vec{x} \in A^{n_\phi}, w \in A_{s_\phi}, \text{ and } \phi^{\mathbf{A}}(\vec{x}) = w\}$$

- We may assume that $\mathbf{A}$ is completely determined by $\mathrm{eqdiag}(\mathbf{A})$

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

($\star$) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- ($\star$) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- ($\star$) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

($\star$) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \le \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- ($\star$) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- ($\star$) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

($\star$) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \le \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- ($\star$) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- ($\star$) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \text{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

$(\star)$ If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\text{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \text{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- $(\star)$ is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- $(\star)$ is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

$(\star)$ If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- $(\star)$ is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- $(\star)$ is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

(⋆) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- (⋆) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- (⋆) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \text{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

($\star$) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\text{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \text{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- ($\star$) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- ($\star$) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Sample result: the intrinsic calls complexity

With each structure $\mathbf{A} = (A, \mathbf{\Phi})$, each $\Phi_0 \subseteq \Phi$ and each (partial) function or relation $f : A^n \rightharpoonup A_s$ we will associate a partial function

$$\boxed{\vec{x} \mapsto \mathsf{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \in \mathbb{N} \qquad (f(\vec{x})\downarrow)}$$

such that:

($\star$) If α is any algorithm from $\mathbf{\Phi}$ which computes f, then

$$\mathsf{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) \leq \mathsf{calls}_{\Phi_0}(\alpha, \vec{x}) \quad (f(\vec{x})\downarrow)$$

- ($\star$) is not trivial: in some important examples in arithmetic and algebra it yields the best known lower bound results
- ($\star$) is a theorem for concrete algorithms specified by the usual computation models; it is plausible for all algorithms from $\mathbf{\Phi}$
- The results are about several natural complexity measures on algorithms from primitives, not only "the number of calls to Φ_0"
- The methods are from abstract model theory

Slogan: *Absolute lower bound results*
are the undecidability facts about decidable problems

(1) Preliminaries
(2) Uniform processes
(3) Comprimeness in $\mathbb{N}$
(4) Polynomial 0-testing

Is the Euclidean algorithm optimal among its peers? (with vDD, 2004)
Arithmetic complexity (with van Den Dries, 2009)
Recursion and complexity (notes) `www.math.ucla.edu/~ynm`
(currently under repair)

Y. Mansour, B. Schieber, and P. Tiwari (1991)
A lower bound for integer greatest common divisor computations
Lower bounds for computations with the floor operation
J. Meidânis (1991): *Lower bounds for arithmetic problems*
P. Bürgisser and T. Lickteig (1992)
Verification complexity of linear prime ideals
P. Bürgisser, T. Lickteig, and M. Shub (1992),
Test complexity of generic polynomials

Slogan: *Absolute lower bound results*
are the undecidability facts about decidable problems

(1) Preliminaries
(2) Uniform processes
(3) Comprimeness in $\mathbb{N}$
(4) Polynomial 0-testing

Is the Euclidean algorithm optimal among its peers? (with vDD, 2004)
Arithmetic complexity (with van Den Dries, 2009)
Recursion and complexity (notes) `www.math.ucla.edu/∼ynm`
(currently under repair)

Y. Mansour, B. Schieber, and P. Tiwari (1991)
A lower bound for integer greatest common divisor computations
Lower bounds for computations with the floor operation
J. Meidânis (1991): *Lower bounds for arithmetic problems*
P. Bürgisser and T. Lickteig (1992)
Verification complexity of linear prime ideals
P. Bürgisser, T. Lickteig, and M. Shub (1992),
Test complexity of generic polynomials

Slogan: *Absolute lower bound results*
are the undecidability facts about decidable problems

(1) Preliminaries
(2) Uniform processes
(3) Comprimeness in $\mathbb{N}$
(4) Polynomial 0-testing

Is the Euclidean algorithm optimal among its peers? (with vDD, 2004)
Arithmetic complexity (with van Den Dries, 2009)
Recursion and complexity (notes) `www.math.ucla.edu/∼ynm`
(currently under repair)

Y. Mansour, B. Schieber, and P. Tiwari (1991)
A lower bound for integer greatest common divisor computations
Lower bounds for computations with the floor operation
J. Meidânis (1991): *Lower bounds for arithmetic problems*
P. Bürgisser and T. Lickteig (1992)
Verification complexity of linear prime ideals
P. Bürgisser, T. Lickteig, and M. Shub (1992),
Test complexity of generic polynomials

Slogan: *Absolute lower bound results*
are the undecidability facts about decidable problems

(1) Preliminaries
(2) Uniform processes
(3) Comprimeness in $\mathbb{N}$
(4) Polynomial 0-testing

Is the Euclidean algorithm optimal among its peers? (with vDD, 2004)
Arithmetic complexity (with van Den Dries, 2009)
Recursion and complexity (notes) `www.math.ucla.edu/~ynm`
(currently under repair)

Y. Mansour, B. Schieber, and P. Tiwari (1991)
A lower bound for integer greatest common divisor computations
Lower bounds for computations with the floor operation
J. Meidânis (1991): *Lower bounds for arithmetic problems*
P. Bürgisser and T. Lickteig (1992)
Verification complexity of linear prime ideals
P. Bürgisser, T. Lickteig, and M. Shub (1992),
Test complexity of generic polynomials

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \mathbf{\Phi}) \iff U \subseteq A \;\&\; \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \;\&\; (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

Substructures may be finite and not closed under $\mathbf{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \ldots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \ldots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \ldots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \mathbf{\Phi}) \iff U \subseteq A \ \& \ \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \ \& \ (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

Substructures may be finite and not closed under $\mathbf{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \ldots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \ldots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \ldots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \boldsymbol{\Phi}) \iff U \subseteq A \;\&\; \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \;\&\; (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

Substructures may be finite and not closed under $\boldsymbol{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \ldots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \ldots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \ldots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \mathbf{\Phi}) \iff U \subseteq A \ \& \ \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \ \& \ (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

  Substructures may be finite and not closed under $\mathbf{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \dots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \dots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \dots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \mathbf{\Phi}) \iff U \subseteq A \;\&\; \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \;\&\; (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

Substructures may be finite and not closed under $\mathbf{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \ldots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \ldots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \ldots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Substructures and homomorphisms

- Substructures (pieces):

$$\mathbf{U} \subseteq_p \mathbf{A} = (A, \mathbf{\Phi}) \iff U \subseteq A \ \&\ \mathrm{eqdiag}(\mathbf{U}) \subseteq \mathrm{eqdiag}(\mathbf{A})$$
$$\iff U \subseteq A \ \&\ (\forall \phi \in \Phi)[\phi^{\mathbf{U}} \sqsubseteq \phi^{\mathbf{A}}]$$

Substructures may be finite and not closed under $\mathbf{\Phi}$

- A homomorphism $\pi : \mathbf{U} \rightarrowtail \mathbf{V}$ is any $\pi : U \to V$ such that for all $\phi \in \Phi, x_1, \ldots, x_n \in U, w \in U_s$, (with $\pi(\mathrm{tt}) = \mathrm{tt}, \pi(\mathrm{ff}) = \mathrm{ff}$)

$$\phi^{\mathbf{U}}(x_1, \ldots, x_n) = w \implies \phi^{\mathbf{V}}(\pi x_1, \ldots, \pi x_n) = \pi w$$

  - May have $x \neq y, \pi(x) = \pi(y)$, unless $(=, x, y, \mathrm{ff}) \in \mathrm{eqdiag}(\mathbf{U})$
  - π is an embedding if it is injective (in which case it preserves $\neq$)

- We use finite substructures $\mathbf{U} \subseteq_p \mathbf{A}$ to represent calls to the primitives executed during a computation in $\mathbf{A}$

Algorithms from primitives – the basic intuition

An n-ary algorithm α of $\mathbf{A} = (A, \mathbf{\Phi})$ (or from $\mathbf{\Phi}$) "computes" some n-ary partial function or relation

$$\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$$

using the primitives in $\mathbf{\Phi}$ as oracles and nothing else about $\mathbf{A}$

We understand this to mean that in the course of a "computation" of $\overline{\alpha}(\vec{x})$, the algorithm may request from the oracle for any $\phi^{\mathbf{A}}$ any particular value $\phi^{\mathbf{A}}(\vec{u})$, for arguments $\vec{u}$ which it has already computed from $\vec{x}$, and that if the oracles cooperate, then "the computation" of $\overline{\alpha}(\vec{x})$ is completed in a finite number of "steps"

- The notion of a uniform process attempts to capture minimally (in the style of abstract model theory) these aspects of algorithms from primitives
- It does not capture their effectiveness, but their uniformity —that an algorithm applies "the same procedure" to all arguments in its domain

Algorithms from primitives – the basic intuition

An n-ary algorithm α of $\mathbf{A} = (A, \mathbf{\Phi})$ (or from $\mathbf{\Phi}$) "computes" some n-ary partial function or relation

$$\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$$

using the primitives in $\mathbf{\Phi}$ as oracles and nothing else about $\mathbf{A}$

We understand this to mean that in the course of a "computation" of $\overline{\alpha}(\vec{x})$, the algorithm may request from the oracle for any $\phi^{\mathbf{A}}$ any particular value $\phi^{\mathbf{A}}(\vec{u})$, for arguments $\vec{u}$ which it has already computed from $\vec{x}$, and that if the oracles cooperate, then "the computation" of $\overline{\alpha}(\vec{x})$ is completed in a finite number of "steps"

- The notion of a uniform process attempts to capture minimally (in the style of abstract model theory) these aspects of algorithms from primitives
- It does not capture their effectiveness, but their uniformity —that an algorithm applies "the same procedure" to all arguments in its domain

Algorithms from primitives – the basic intuition

An n-ary algorithm α of $\mathbf{A} = (A, \mathbf{\Phi})$ (or from $\mathbf{\Phi}$) "computes" some n-ary partial function or relation

$$\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$$

using the primitives in $\mathbf{\Phi}$ as oracles and nothing else about $\mathbf{A}$

We understand this to mean that in the course of a "computation" of $\overline{\alpha}(\vec{x})$, the algorithm may request from the oracle for any $\phi^{\mathbf{A}}$ any particular value $\phi^{\mathbf{A}}(\vec{u})$, for arguments $\vec{u}$ which it has already computed from $\vec{x}$, and that if the oracles cooperate, then "the computation" of $\overline{\alpha}(\vec{x})$ is completed in a finite number of "steps"

- The notion of a uniform process attempts to capture minimally (in the style of abstract model theory) these aspects of algorithms from primitives
- It does not capture their effectiveness, but their uniformity —that an algorithm applies "the same procedure" to all arguments in its domain

Algorithms from primitives – the basic intuition

An n-ary algorithm α of $\mathbf{A} = (A, \mathbf{\Phi})$ (or from $\mathbf{\Phi}$) "computes" some n-ary partial function or relation

$$\overline{\alpha} = \overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$$

using the primitives in $\mathbf{\Phi}$ as oracles and nothing else about $\mathbf{A}$

We understand this to mean that in the course of a "computation" of $\overline{\alpha}(\vec{x})$, the algorithm may request from the oracle for any $\phi^{\mathbf{A}}$ any particular value $\phi^{\mathbf{A}}(\vec{u})$, for arguments $\vec{u}$ which it has already computed from $\vec{x}$, and that if the oracles cooperate, then "the computation" of $\overline{\alpha}(\vec{x})$ is completed in a finite number of "steps"

- The notion of a uniform process attempts to capture minimally (in the style of abstract model theory) these aspects of algorithms from primitives
- It does not capture their effectiveness, but their uniformity —that an algorithm applies "the same procedure" to all arguments in its domain

Uniform processes: I The Locality Axiom

A *uniform process* α *of arity* n *and sort* s *of a structure* $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ *assigns to each substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ *an* n*-ary partial function*

$$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$

It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$

- For an algorithm α, intuitively, $\overline{\alpha}^{\mathbf{U}}$ is the restriction to U of the partial function computed by α when the oracles respond only to questions with answers in eqdiag($\mathbf{U}$)

We write

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \iff \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w,$$

$$\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff (\exists w)[\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w]$$

Uniform processes: I The Locality Axiom

A uniform process α of arity n and sort s of a structure $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ assigns to each substructure $\mathbf{U} \subseteq_p \mathbf{A}$ an n-ary partial function

$$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$

It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$

- For an algorithm α, intuitively, $\overline{\alpha}^{\mathbf{U}}$ is the restriction to U of the partial function computed by α when the oracles respond only to questions with answers in eqdiag($\mathbf{U}$)

We write

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \iff \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w,$$

$$\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff (\exists w)[\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w]$$

Uniform processes: I The Locality Axiom

A *uniform process* α *of arity n and sort s of a structure* $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ *assigns to each substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ *an n-ary partial function*

$$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$

It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$

- For an algorithm α, intuitively, $\overline{\alpha}^{\mathbf{U}}$ is the restriction to U of the partial function computed by α when the oracles respond only to questions with answers in eqdiag($\mathbf{U}$)

We write

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \iff \overline{\alpha}^{\mathbf{U}}(\vec{x}) = w,$$

$$\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff (\exists w)[\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w]$$

Uniform processes: II The Homomorphism Axiom

If α is an n-ary uniform process of $\mathbf{A}$, $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$, and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism, then

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \implies \mathbf{V} \vdash \alpha(\pi\vec{x}) = \pi w \quad (x_1, \dots, x_n \in U, w \in U_s)$$

In particular, if $\mathbf{U} \subseteq_p \mathbf{A}$, then $\overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{A}}$

- For algorithms: when asked for $\phi^{\mathbf{U}}(\vec{x})$, the oracle for ϕ may consistently provide $\phi^{\mathbf{V}}(\pi\vec{x})$, if π is a homomorphism
- This is obvious for the identity embedding $I : \mathbf{U} \rightarrowtail \mathbf{A}$, but it is a strong restriction for algorithms from rich primitives (stacks, higher type constructs, etc.)
- It can be verified for the standard computation models (deterministic and non-deterministic) provided all their primitives are included in $\mathbf{\Phi}$

Uniform processes: II The Homomorphism Axiom

If α is an n-ary uniform process of $\mathbf{A}$, $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$, *and* $\pi : \mathbf{U} \to \mathbf{V}$ *is a homomorphism, then*

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \implies \mathbf{V} \vdash \alpha(\pi\vec{x}) = \pi w \quad (x_1, \ldots, x_n \in U, w \in U_s)$$

In particular, if $\mathbf{U} \subseteq_p \mathbf{A}$, then $\overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{A}}$

- For algorithms: when asked for $\phi^{\mathbf{U}}(\vec{x})$, the oracle for ϕ may consistently provide $\phi^{\mathbf{V}}(\pi\vec{x})$, if π is a homomorphism
- This is obvious for the identity embedding $I : \mathbf{U} \rightarrowtail \mathbf{A}$, but it is a strong restriction for algorithms from rich primitives (stacks, higher type constructs, etc.)
- It can be verified for the standard computation models (deterministic and non-deterministic) provided all their primitives are included in $\mathbf{\Phi}$

Uniform processes: II The Homomorphism Axiom

If α is an n-ary uniform process of $\mathbf{A}$, $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$, and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism, then

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \implies \mathbf{V} \vdash \alpha(\pi\vec{x}) = \pi w \quad (x_1, \ldots, x_n \in U, w \in U_s)$$

In particular, if $\mathbf{U} \subseteq_p \mathbf{A}$, then $\overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{A}}$

- For algorithms: when asked for $\phi^{\mathbf{U}}(\vec{x})$, the oracle for ϕ may consistently provide $\phi^{\mathbf{V}}(\pi\vec{x})$, if π is a homomorphism
- This is obvious for the identity embedding $I : \mathbf{U} \rightarrowtail \mathbf{A}$, but it is a strong restriction for algorithms from rich primitives (stacks, higher type constructs, etc.)
- It can be verified for the standard computation models (deterministic and non-deterministic) provided all their primitives are included in $\mathbf{\Phi}$

Uniform processes: II The Homomorphism Axiom

If α is an n-ary uniform process of $\mathbf{A}$, $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$, and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism, then

$$\mathbf{U} \vdash \alpha(\vec{x}) = w \implies \mathbf{V} \vdash \alpha(\pi\vec{x}) = \pi w \quad (x_1, \ldots, x_n \in U, w \in U_s)$$

In particular, if $\mathbf{U} \subseteq_p \mathbf{A}$, then $\overline{\alpha}^{\mathbf{U}} \sqsubseteq \overline{\alpha}^{\mathbf{A}}$

- For algorithms: when asked for $\phi^{\mathbf{U}}(\vec{x})$, the oracle for ϕ may consistently provide $\phi^{\mathbf{V}}(\pi\vec{x})$, if π is a homomorphism
- This is obvious for the identity embedding $I : \mathbf{U} \rightarrowtail \mathbf{A}$, but it is a strong restriction for algorithms from rich primitives (stacks, higher type constructs, etc.)
- It can be verified for the standard computation models (deterministic and non-deterministic) provided all their primitives are included in $\mathbf{\Phi}$

Uniform processes: III The Finiteness Axiom

If α is an n-ary uniform process of $\mathbf{A}$*, then*

$$\begin{aligned}&\mathbf{A} \vdash \alpha(\vec{x}) = w\\ \Longrightarrow\ &\textit{there is a finite } \mathbf{U} \subseteq_p \mathbf{A} \textit{ generated by } \vec{x} \textit{ such that } \mathbf{U} \vdash \alpha(\vec{x}) = w\end{aligned}$$

- For every call $\phi(\vec{u})$ to the primitives, the algorithm must construct the arguments $\vec{u}$, and so the entire computation takes place within a finite substructure generated by the input $\vec{x}$

We write

$$\mathbf{U} \vdash_c \alpha(\vec{x}) = w \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \vdash \alpha(\vec{x}) = w,$$

$$\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff (\exists w)[\mathbf{U} \vdash_c \alpha(\vec{x}) = w]$$

and we think of $(\mathbf{U}, \vec{x}, w)$ as a computation of α on the input $\vec{x}$

Uniform processes: III The Finiteness Axiom

If α is an n-ary uniform process of $\mathbf{A}$*, then*

$$\mathbf{A} \vdash \alpha(\vec{x}) = w$$
$$\Longrightarrow \textit{ there is a finite } \mathbf{U} \subseteq_p \mathbf{A} \textit{ generated by } \vec{x} \textit{ such that } \mathbf{U} \vdash \alpha(\vec{x}) = w$$

- For every call $\phi(\vec{u})$ to the primitives, the algorithm must construct the arguments $\vec{u}$, and so the entire computation takes place within a finite substructure generated by the input $\vec{x}$

We write

$$\mathbf{U} \vdash_c \alpha(\vec{x}) = w \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \vdash \alpha(\vec{x}) = w,$$
$$\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff (\exists w)[\mathbf{U} \vdash_c \alpha(\vec{x}) = w]$$

and we think of $(\mathbf{U}, \vec{x}, w)$ as a computation of α on the input $\vec{x}$

Uniform processes: III The Finiteness Axiom

If α is an n-ary uniform process of $\mathbf{A}$, *then*

$$\mathbf{A} \vdash \alpha(\vec{x}) = w$$
$$\Longrightarrow \textit{ there is a finite } \mathbf{U} \subseteq_p \mathbf{A} \textit{ generated by } \vec{x} \textit{ such that } \mathbf{U} \vdash \alpha(\vec{x}) = w$$

- For every call $\phi(\vec{u})$ to the primitives, the algorithm must construct the arguments $\vec{u}$, and so the entire computation takes place within a finite substructure generated by the input $\vec{x}$

We write

$$\mathbf{U} \vdash_c \alpha(\vec{x}) = w \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \vdash \alpha(\vec{x}) = w,$$
$$\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff (\exists w)[\mathbf{U} \vdash_c \alpha(\vec{x}) = w]$$

and we think of $(\mathbf{U}, \vec{x}, w)$ as a computation of α on the input $\vec{x}$

Uniform processes, summary

- I The Locality Axiom:
  A uniform process α of arity n and sort s of a structure $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ assigns to each substructure $\mathbf{U} \subseteq_p \mathbf{A}$ an n-ary partial function

$$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$

  It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$

$$\boxed{\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff \alpha^{\mathbf{U}}(\vec{x})\downarrow}$$

- II The Homomorphism Axiom:
  If $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$ and $\pi : \mathbf{U} \to \mathbf{V}$ is a homomorphism, then

$$\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \implies \overline{\alpha}^{\mathbf{V}}(\pi\vec{x}) = \pi w$$

$$\boxed{\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \overline{\alpha}^{\mathbf{U}}(\vec{x})\downarrow}$$

- III The Finiteness Axiom:

$$\mathbf{A} \vdash \alpha(\vec{x})\downarrow \implies (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow]$$

Uniform processes, summary

- I The Locality Axiom:
  A uniform process α *of arity* n *and sort* s *of a structure* $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ *assigns to each substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ *an* n*-ary partial function*
  $$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$
  It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$
  $$\boxed{\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff \alpha^{\mathbf{U}}(\vec{x})\downarrow}$$

- II The Homomorphism Axiom:
  If $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$ *and* $\pi : \mathbf{U} \to \mathbf{V}$ *is a homomorphism, then*
  $$\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \implies \overline{\alpha}^{\mathbf{V}}(\pi\vec{x}) = \pi w$$
  $$\boxed{\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \overline{\alpha}^{\mathbf{U}}(\vec{x})\downarrow}$$

- III The Finiteness Axiom:
  $$\mathbf{A} \vdash \alpha(\vec{x})\downarrow \implies (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow]$$

Uniform processes, summary

- I The Locality Axiom:
  A uniform process α *of arity n and sort s of a structure* $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ *assigns to each substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ *an n-ary partial function*
  $$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$
  It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$
  $$\boxed{\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff \alpha^{\mathbf{U}}(\vec{x})\downarrow}$$
- II The Homomorphism Axiom:
  If $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$ *and* $\pi : \mathbf{U} \to \mathbf{V}$ *is a homomorphism, then*
  $$\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \implies \overline{\alpha}^{\mathbf{V}}(\pi\vec{x}) = \pi w$$
  $$\boxed{\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \overline{\alpha}^{\mathbf{U}}(\vec{x})\downarrow}$$
- III The Finiteness Axiom:
  $$\mathbf{A} \vdash \alpha(\vec{x})\downarrow \implies (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow]$$

Uniform processes, summary

- I The Locality Axiom:
  A uniform process α *of arity* n *and sort* s *of a structure* $\mathbf{A} = (A, \Phi^{\mathbf{A}})$ *assigns to each substructure* $\mathbf{U} \subseteq_p \mathbf{A}$ *an* n*-ary partial function*

  $$\overline{\alpha}^{\mathbf{U}} : U^n \rightharpoonup U_s$$

  It computes the partial function or relation $\overline{\alpha}^{\mathbf{A}} : A^n \rightharpoonup A_s$

  $$\boxed{\mathbf{U} \vdash \alpha(\vec{x})\downarrow \iff \alpha^{\mathbf{U}}(\vec{x})\downarrow}$$

- II The Homomorphism Axiom:
  If $\mathbf{U}, \mathbf{V} \subseteq_p \mathbf{A}$ *and* $\pi : \mathbf{U} \to \mathbf{V}$ *is a homomorphism, then*

  $$\overline{\alpha}^{\mathbf{U}}(\vec{x}) = w \implies \overline{\alpha}^{\mathbf{V}}(\pi\vec{x}) = \pi w$$

  $$\boxed{\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \overline{\alpha}^{\mathbf{U}}(\vec{x})\downarrow}$$

- III The Finiteness Axiom:

  $$\mathbf{A} \vdash \alpha(\vec{x})\downarrow \implies (\exists \mathbf{U} \subseteq_p \mathbf{A})[\mathbf{U} \vdash_c \alpha(\vec{x})\downarrow]$$

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$

- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$

- $\mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)

- $\mathrm{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)

- $\mathrm{depth}(\alpha, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\mathrm{depth}(\alpha, \vec{x}) \leq \mathrm{size}(\alpha, \vec{x}) \leq \mathrm{calls}(\alpha, \vec{x}) \quad (= \mathrm{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$
- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$
- $\mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)
- $\mathrm{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)
- $\mathrm{depth}(\alpha, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\mathrm{depth}(\alpha, \vec{x}) \leq \mathrm{size}(\alpha, \vec{x}) \leq \mathrm{calls}(\alpha, \vec{x}) \quad (= \mathrm{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$

- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\!\downarrow\}}$

- $\mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\!\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)

- $\mathrm{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\!\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)

- $\mathrm{depth}(\alpha, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\!\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\mathrm{depth}(\alpha, \vec{x}) \leq \mathrm{size}(\alpha, \vec{x}) \leq \mathrm{calls}(\alpha, \vec{x}) \quad (= \mathrm{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$

- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$

- $\mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
(the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)

- $\mathrm{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
(the least number of elements of $\mathbf{A}$ that α must see)

- $\mathrm{depth}(\alpha, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
(the least number of calls α must execute in sequence)

Thm $\boxed{\mathrm{depth}(\alpha, \vec{x}) \le \mathrm{size}(\alpha, \vec{x}) \le \mathrm{calls}(\alpha, \vec{x}) \quad (= \mathrm{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$
- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$
- $\text{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\text{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)
- $\text{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)
- $\text{depth}(\alpha, \vec{x}) = \min\{\text{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\text{depth}(\alpha, \vec{x}) \le \text{size}(\alpha, \vec{x}) \le \text{calls}(\alpha, \vec{x}) \quad (= \text{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$
- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$
- $\text{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\text{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)
- $\text{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)
- $\text{depth}(\alpha, \vec{x}) = \min\{\text{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\text{depth}(\alpha, \vec{x}) \le \text{size}(\alpha, \vec{x}) \le \text{calls}(\alpha, \vec{x}) \quad (= \text{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$
- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$
- $\mathrm{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)
- $\mathrm{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)
- $\mathrm{depth}(\alpha, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\mathrm{depth}(\alpha, \vec{x}) \leq \mathrm{size}(\alpha, \vec{x}) \leq \mathrm{calls}(\alpha, \vec{x}) \quad (= \mathrm{calls}_\Phi(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

Complexity measures for uniform processes

- A substructure norm μ on $\mathbf{A}$ assigns to each finite $\mathbf{U} \subseteq_p \mathbf{A}$ generated by $\vec{x} \in U^n$ a number $\mu(\mathbf{U}, \vec{x})$
- $\boxed{C_\mu(\alpha, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}}$
- $\text{calls}_{\Phi_0}(\alpha, \vec{x}) = \min\{|\text{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\} \quad (\Phi_0 \subseteq \Phi)$
  (the least number of calls to $\phi \in \Phi_0$ α must do to compute $\overline{\alpha}^{\mathbf{A}}(\vec{x})$)
- $\text{size}(\alpha, \vec{x}) = \min\{|U| : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of elements of $\mathbf{A}$ that α must see)
- $\text{depth}(\alpha, \vec{x}) = \min\{\text{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \vdash_c \alpha(\vec{x})\downarrow\}$
  (the least number of calls α must execute in sequence)

Thm $\boxed{\text{depth}(\alpha, \vec{x}) \le \text{size}(\alpha, \vec{x}) \le \text{calls}(\alpha, \vec{x}) \quad (= \text{calls}_{\Phi}(\alpha, \vec{x}))}$

These are not larger than standard definitions for concrete algorithms

★ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, $\mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \ \&\ f(\vec{x}) \in U_s \ \&\ \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \ \&\ \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{A} \text{ respects } f \text{ at } \vec{x}$$

$$\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$$

$$\boxed{\text{The intrinsic complexities of } f \text{ in } \mathbf{A}}$$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$

$\star$ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s,\ f(\vec{x})\downarrow,\ \mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \ \&\ f(\vec{x}) \in U_s \ \&\ \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \ \&\ \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{A} \text{ respects } f \text{ at } \vec{x}$$

$$\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$$

The intrinsic complexities of f in $\mathbf{A}$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$

⋆ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, $\mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \;\&\; f(\vec{x}) \in U_s \;\&\; \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \;\&\; \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff$ every homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$

$\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff \mathbf{U}$ is finite, generated by $\vec{x}$ and $\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$

$\boxed{\text{The intrinsic complexities of } f \text{ in } \mathbf{A}}$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$

⋆ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, $\mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \;\&\; f(\vec{x}) \in U_s \;\&\; \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \;\&\; \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{A} \text{ respects } f \text{ at } \vec{x}$$

$$\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$$

$\boxed{\text{The intrinsic complexities of } f \text{ in } \mathbf{A}}$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \upharpoonright \Phi_0)| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$

$\star$ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, $\mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \ \& \ f(\vec{x}) \in U_s \ \& \ \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \ \& \ \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff$ every homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$

$\mathbf{U} \Vdash_c^{\mathbf{A}} f(\vec{x})\downarrow \iff$ $\mathbf{U}$ is finite, generated by $\vec{x}$ and $\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$

$\boxed{\text{The intrinsic complexities of } f \text{ in } \mathbf{A}}$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \Vdash_c^{\mathbf{A}} f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash_c^{\mathbf{A}} f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c^{\mathbf{A}} f(\vec{x})\downarrow\}$

⋆ The forcing and certification relations

Suppose $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, $\mathbf{U} \subseteq_p \mathbf{A}$.

- A homomorphism $\pi : \mathbf{U} \to \mathbf{A}$ respects f at $\vec{x}$ if

$$\boxed{\vec{x} \in U^n \;\&\; f(\vec{x}) \in U_s \;\&\; \pi(f(\vec{x})) = f(\pi(\vec{x}))}$$

so for relations $\boxed{\vec{x} \in U^n \;\&\; \Big(R(\vec{x}) \iff R(\pi(\vec{x}))\Big)}$

$$\mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow \iff \text{every homomorphism } \pi : \mathbf{U} \to \mathbf{A} \text{ respects } f \text{ at } \vec{x}$$

$$\mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow \iff \mathbf{U} \text{ is finite, generated by } \vec{x} \text{ and } \mathbf{U} \Vdash^{\mathbf{A}} f(\vec{x})\downarrow$$

The intrinsic complexities of f in $\mathbf{A}$

- $C_\mu(\mathbf{A}, f, \vec{x}) = \min\{\mu(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash_c f(\vec{x})\downarrow\} \in \mathbb{N} \cup \{\infty\}$
- $\mathrm{calls}_{\Phi_0}(\mathbf{A}, f, \vec{x}) = \min\{|\mathrm{eqdiag}(\mathbf{U} \restriction \Phi_0)| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{size}(\mathbf{A}, f, \vec{x}) = \min\{|U| : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$
- $\mathrm{depth}(\mathbf{A}, f, \vec{x}) = \min\{\mathrm{depth}(\mathbf{U}, \vec{x}) : \mathbf{U} \Vdash^{\mathbf{A}}_c f(\vec{x})\downarrow\}$

Deriving lower bounds by constructing homomorphisms

- The following two facts are immediate from the definitions:

Lemma

If α is a uniform process which computes $f : A^n \rightharpoonup A_s$ in $\mathbf{A}$, then

$$C_\mu(\mathbf{A}, f, \vec{x}) \leq C_\mu(\alpha, \vec{x}) \qquad (f(\vec{x})\downarrow)$$

Lemma (The homomorphism test)

Suppose μ is a substructure norm (e.g., $\mathrm{calls}_{\Phi_0}, \mathrm{size}, \mathrm{depth}$) on a Φ-structure $\mathbf{A}$, $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, and

for every finite $\mathbf{U} \subseteq_p \mathbf{A}$ which is generated by $\vec{x}$,

$$\Big(f(\vec{x}) \in U_s \;\&\; \mu(\mathbf{U}, \vec{x}) < m\Big) \implies (\exists \pi : \mathbf{U} \to \mathbf{A})[f(\pi(\vec{x})) \neq \pi(f(\vec{x}))];$$

then $C_\mu(\mathbf{A}, f, \vec{x}) \geq m$.

Deriving lower bounds by constructing homomorphisms

- The following two facts are immediate from the definitions:

Lemma

If α is a uniform process which computes $f : A^n \rightharpoonup A_s$ in $\mathbf{A}$, then

$$C_\mu(\mathbf{A}, f, \vec{x}) \le C_\mu(\alpha, \vec{x}) \qquad (f(\vec{x})\downarrow)$$

Lemma (The homomorphism test)

Suppose μ is a substructure norm (e.g., $\mathrm{calls}_{\Phi_0}, \mathrm{size}, \mathrm{depth}$) on a Φ-structure $\mathbf{A}$, $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, and

for every finite $\mathbf{U} \subseteq_p \mathbf{A}$ which is generated by $\vec{x}$,

$$\Big(f(\vec{x}) \in U_s \;\&\; \mu(\mathbf{U}, \vec{x}) < m\Big) \implies (\exists \pi : \mathbf{U} \to \mathbf{A})[f(\pi(\vec{x})) \neq \pi(f(\vec{x}))];$$

then $C_\mu(\mathbf{A}, f, \vec{x}) \ge m$.

Deriving lower bounds by constructing homomorphisms

- The following two facts are immediate from the definitions:

Lemma

If α is a uniform process which computes $f : A^n \rightharpoonup A_s$ in $\mathbf{A}$, then

$$C_\mu(\mathbf{A}, f, \vec{x}) \le C_\mu(\alpha, \vec{x}) \qquad (f(\vec{x})\downarrow)$$

Lemma (The homomorphism test)

Suppose μ is a substructure norm (e.g., $\mathrm{calls}_{\Phi_0}, \mathrm{size}, \mathrm{depth}$) on a Φ-structure $\mathbf{A}$, $f : A^n \rightharpoonup A_s$, $f(\vec{x})\downarrow$, and

for every finite $\mathbf{U} \subseteq_p \mathbf{A}$ which is generated by $\vec{x}$,

$$\Big(f(\vec{x}) \in U_s \;\&\; \mu(\mathbf{U}, \vec{x}) < m\Big) \implies (\exists \pi : \mathbf{U} \to \mathbf{A})[f(\pi(\vec{x})) \ne \pi(f(\vec{x}))];$$

then $C_\mu(\mathbf{A}, f, \vec{x}) \ge m$.

A lower bound for coprimeness on $\mathbb{N}$

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$, $\mathbf{\Psi}$ a finite set of *Presburger functions*

Theorem (van den Dries, ynm, 2004, 2009)

If $\xi > 1$ is quadratic irrational, then for some $r > 0$ and all sufficiently large coprime (a, b),

$$\left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}(\mathbf{A}, \perp, a, b) \geq r \log\log a. \qquad (1)$$

In particular, the conclusion of (1) holds with some r

- *for positive Pell pairs (a, b) satisfying $a^2 = 2b^2 + 1$* $\quad (\xi = \sqrt{2})$
- *for Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$* $\quad (\xi = \frac{1}{2}(1 + \sqrt{5}))$

Theorem (Pratt, unpublished)

There is a non-deterministic algorithm ε_{nd} of $\mathbf{N}_\varepsilon$ which decides coprimeness, is at least as effective as the Euclidean everywhere and

$$\mathrm{calls}(\varepsilon_{nd}, F_{k+1}, F_k) \leq K \log\log F_{k+1}$$

- The theorem is best possible from its hypotheses

A lower bound for coprimeness on $\mathbb{N}$

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \boldsymbol{\Psi})$, $\boldsymbol{\Psi}$ a finite set of *Presburger functions*

Theorem (van den Dries, ynm, 2004, 2009)

If $\xi > 1$ is quadratic irrational, then for some $r > 0$ and all sufficiently large coprime (a, b),

$$\left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, a, b) \geq r \log\log a. \qquad (1)$$

In particular, the conclusion of (1) holds with some r

- *for positive Pell pairs (a, b) satisfying $a^2 = 2b^2 + 1$ $\quad (\xi = \sqrt{2})$*
- *for Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$ $\quad (\xi = \frac{1}{2}(1 + \sqrt{5}))$*

Theorem (Pratt, unpublished)

There is a non-deterministic algorithm ε_{nd} of $\mathbf{N}_\varepsilon$ which decides coprimeness, is at least as effective as the Euclidean everywhere and

$$\mathrm{calls}(\varepsilon_{nd}, F_{k+1}, F_k) \leq K \log\log F_{k+1}$$

- The theorem is best possible from its hypotheses

A lower bound for coprimeness on $\mathbb{N}$

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$, $\mathbf{\Psi}$ a finite set of *Presburger functions*

Theorem (van den Dries, ynm, 2004, 2009)

If $\xi > 1$ is quadratic irrational, then for some $r > 0$ and all sufficiently large coprime (a, b),

$$\left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, a, b) \geq r \log\log a. \qquad (1)$$

In particular, the conclusion of (1) holds with some r

- *for positive Pell pairs (a, b) satisfying $a^2 = 2b^2 + 1$ $\quad(\xi = \sqrt{2})$*
- *for Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$ $\quad(\xi = \frac{1}{2}(1 + \sqrt{5}))$*

Theorem (Pratt, unpublished)

There is a non-deterministic algorithm ε_{nd} of $\mathbf{N}_\varepsilon$ which decides coprimeness, is at least as effective as the Euclidean everywhere and

$$\mathrm{calls}(\varepsilon_{nd}, F_{k+1}, F_k) \leq K \log\log F_{k+1}$$

- The theorem is best possible from its hypotheses

A lower bound for coprimeness on $\mathbb{N}$

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$, $\mathbf{\Psi}$ a finite set of *Presburger functions*

Theorem (van den Dries, ynm, 2004, 2009)

If $\xi > 1$ is quadratic irrational, then for some $r > 0$ and all sufficiently large coprime (a, b),

$$\left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, a, b) \geq r \log\log a. \qquad (1)$$

In particular, the conclusion of (1) holds with some r

- *for positive Pell pairs (a, b) satisfying $a^2 = 2b^2 + 1$* $\quad (\xi = \sqrt{2})$
- *for Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$* $\quad (\xi = \frac{1}{2}(1 + \sqrt{5}))$

Theorem (Pratt, unpublished)

There is a non-deterministic algorithm ε_{nd} of $\mathbf{N}_\varepsilon$ which decides coprimeness, is at least as effective as the Euclidean everywhere and

$$\mathrm{calls}(\varepsilon_{nd}, F_{k+1}, F_k) \leq K \log\log F_{k+1}$$

- The theorem is best possible from its hypotheses

A lower bound for coprimeness on $\mathbb{N}$

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$, $\mathbf{\Psi}$ a finite set of *Presburger functions*

Theorem (van den Dries, ynm, 2004, 2009)

If $\xi > 1$ is quadratic irrational, then for some $r > 0$ and all sufficiently large coprime (a, b),

$$\left|\xi - \frac{a}{b}\right| < \frac{1}{b^2} \implies \mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, a, b) \geq r \log\log a. \qquad (1)$$

In particular, the conclusion of (1) holds with some r

- *for positive Pell pairs (a, b) satisfying $a^2 = 2b^2 + 1$ $\quad (\xi = \sqrt{2})$*
- *for Fibonacci pairs (F_{k+1}, F_k) with $k \geq 3$ $\quad (\xi = \frac{1}{2}(1 + \sqrt{5}))$*

Theorem (Pratt, unpublished)

There is a non-deterministic algorithm ε_{nd} of $\mathbf{N}_\varepsilon$ which decides coprimeness, is at least as effective as the Euclidean everywhere and

$$\mathrm{calls}(\varepsilon_{nd}, F_{k+1}, F_k) \leq K \log\log F_{k+1}$$

- The theorem is best possible from its hypotheses

Non-uniform complexity

Given N, how good can a coprimeness algorithm be if we only insist that it works for and uses only N-bit numbers?

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$ as before.
For any N, and any one of the intrinsic complexities as above, let

$$C_\mu(\mathbf{A}, f, 2^N) = \max\{\, C_\mu(\mathbf{A} \upharpoonright [0, 2^N), f, \vec{x}) : x_1, \ldots, x_n < 2^N \}$$

Theorem (van den Dries, ynm 2009)

For some rational number $r > 0$ and all sufficiently large N,

$$\mathrm{calls}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq \mathrm{size}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq r \log N.$$

- Non-uniform lower bound for $\mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, 2^N)$?

Non-uniform complexity

Given N, how good can a coprimeness algorithm be if we only insist that it works for and uses only N-bit numbers?

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \mathbf{\Psi})$ as before.
For any N, and any one of the intrinsic complexities as above, let

$$C_\mu(\mathbf{A}, f, 2^N) = \max\{C_\mu(\mathbf{A} \upharpoonright [0, 2^N), f, \vec{x}) : x_1, \ldots, x_n < 2^N\}$$

Theorem (van den Dries, ynm 2009)

For some rational number $r > 0$ and all sufficiently large N,

$$\mathrm{calls}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq \mathrm{size}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq r \log N.$$

- Non-uniform lower bound for $\mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, 2^N)$?

Non-uniform complexity

Given N, how good can a coprimeness algorithm be if we only insist that it works for and uses only N-bit numbers?

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \boldsymbol{\Psi})$ as before.
For any N, and any one of the intrinsic complexities as above, let

$$C_\mu(\mathbf{A}, f, 2^N) = \max\{C_\mu(\mathbf{A} \restriction [0, 2^N), f, \vec{x}) : x_1, \dots, x_n < 2^N\}$$

Theorem (van den Dries, ynm 2009)

For some rational number $r > 0$ and all sufficiently large N,

$$\mathrm{calls}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq \mathrm{size}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq r \log N.$$

- Non-uniform lower bound for $\mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, 2^N)$?

Non-uniform complexity

Given N, how good can a coprimeness algorithm be if we only insist that it works for and uses only N-bit numbers?

$\mathbf{A} = (\mathbb{N}, 0, 1, +, \dot{-}, \mathrm{iq}, \mathrm{rem}, =, <, \boldsymbol{\Psi})$ as before.
For any N, and any one of the intrinsic complexities as above, let

$$C_\mu(\mathbf{A}, f, 2^N) = \max\{C_\mu(\mathbf{A} \restriction [0, 2^N), f, \vec{x}) : x_1, \dots, x_n < 2^N\}$$

Theorem (van den Dries, ynm 2009)

For some rational number $r > 0$ and all sufficiently large N,

$$\mathrm{calls}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq \mathrm{size}(\mathbf{A}, \perp\!\!\!\perp, 2^N) \geq r \log N.$$

- Non-uniform lower bound for $\mathrm{depth}(\mathbf{A}, \perp\!\!\!\perp, 2^N)$?

The optimality of Horner's rule for polynomial 0-testing

The nullity relation on a field F:

$$N_F(a_0, \ldots, a_n, x) \iff a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

Theorem

Let F be the field of real or complex numbers.
If $n \geq 1$ and $a_0, \ldots, a_n, x$ are algebraically independent in F, then:

(1) $\text{calls}_{\{\cdot,\div\}}(F, N_F, \vec{a}, x) = n$

(2) $\text{calls}_{\{\cdot,\div,=\}}(F, N_F, \vec{a}, x) = n + 1$

- The method for constructing the required homomorphsms is an elaboration of Winograd's proof of the optimality of Horner's rule for poly evaluation
- It is quite different from the method used in arithmetic and requires a homomorphism which is not an embedding in (2)
- Due to Bürgisser and Lickteig (1992) for algebraic decision trees, along with much stronger results

The optimality of Horner's rule for polynomial 0-testing

The nullity relation on a field F:

$$N_F(a_0, \ldots, a_n, x) \iff a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

Theorem

Let F be the field of real or complex numbers.
If $n \geq 1$ and $a_0, \ldots, a_n, x$ are algebraically independent in F, then:

(1) $\text{calls}_{\{\cdot, \div\}}(F, N_F, \vec{a}, x) = n$

(2) $\text{calls}_{\{\cdot, \div, =\}}(F, N_F, \vec{a}, x) = n + 1$

- The method for constructing the required homomorphsms is an elaboration of Winograd's proof of the optimality of Horner's rule for poly evaluation
- It is quite different from the method used in arithmetic and requires a homomorphism which is not an embedding in (2)
- Due to Bürgisser and Lickteig (1992) for algebraic decision trees, along with much stronger results

The optimality of Horner's rule for polynomial 0-testing

The nullity relation on a field F:

$$N_F(a_0, \dots, a_n, x) \iff a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

Theorem

Let F be the field of real or complex numbers.
If $n \geq 1$ and $a_0, \dots, a_n, x$ are algebraically independent in F, then:

(1) $\text{calls}_{\{\cdot,\div\}}(F, N_F, \vec{a}, x) = n$

(2) $\text{calls}_{\{\cdot,\div,=\}}(F, N_F, \vec{a}, x) = n + 1$

- The method for constructing the required homomorphsms is an elaboration of Winograd's proof of the optimality of Horner's rule for poly evaluation
- It is quite different from the method used in arithmetic and requires a homomorphism which is not an embedding in (2)
- Due to Bürgisser and Lickteig (1992) for algebraic decision trees, along with much stronger results

The optimality of Horner's rule for polynomial 0-testing

The nullity relation on a field F:

$$N_F(a_0, \dots, a_n, x) \iff a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

Theorem

Let F be the field of real or complex numbers.
If $n \geq 1$ and $a_0, \dots, a_n, x$ are algebraically independent in F, then:

(1) $\text{calls}_{\{\cdot, \div\}}(F, N_F, \vec{a}, x) = n$

(2) $\text{calls}_{\{\cdot, \div, =\}}(F, N_F, \vec{a}, x) = n + 1$

- The method for constructing the required homomorphsms is an elaboration of Winograd's proof of the optimality of Horner's rule for poly evaluation
- It is quite different from the method used in arithmetic and requires a homomorphism which is not an embedding in (2)
- Due to Bürgisser and Lickteig (1992) for algebraic decision trees, along with much stronger results

The optimality of Horner's rule for polynomial 0-testing

The nullity relation on a field F:

$$N_F(a_0, \dots, a_n, x) \iff a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

Theorem

Let F be the field of real or complex numbers.
If $n \geq 1$ and $a_0, \dots, a_n, x$ are algebraically independent in F, then:

(1) $\text{calls}_{\{\cdot, \div\}}(F, N_F, \vec{a}, x) = n$

(2) $\text{calls}_{\{\cdot, \div, =\}}(F, N_F, \vec{a}, x) = n + 1$

- The method for constructing the required homomorphsms is an elaboration of Winograd's proof of the optimality of Horner's rule for poly evaluation
- It is quite different from the method used in arithmetic and requires a homomorphism which is not an embedding in (2)
- Due to Bürgisser and Lickteig (1992) for algebraic decision trees, along with much stronger results